Distributed Slot Scheduling Algorithm for Hybrid CSMA/TDMA MAC in Wireless Sensor Networks

Manas Ranjan Lenka*, Amulya Ratna Swain* and Manmath Narayanan Sahoo†
* KIIT University, Bhubaneswar, India
†NIT Rourkela, India
Email: {manasy2k3, swainamulya, sahoo.manmath}@gmail.com

***Abstract*—Wireless Sensor Networks(WSNs) consist of many self organized sensor nodes to monitor various activities like temperature, pressure, health condition, intrusion detection, etc. These sensor nodes mostly sense the events happening around them, process the sensed data, and send it to the base station using multiple hops. The base station is connected to the outside world who wants to access these sensed and processed data. In WSN, one of the most important challenge is to handle the collision during data transmission by multiple sensor nodes at the same point of time. The collision during data transmission is handled by proper MAC protocol. The MAC protocols for WSN are broadly categorized into 3 types, i.e. schedule, random, and hybrid. Among these 3 types of MAC protocols, the hybrid MAC protocols try to combine the advantage of both schedule and random based MAC protocols. In this paper, we proposed a distributed slot scheduling algorithm for hybrid MAC algorithm. This algorithm mainly focuses on preparing a schedule which bridges the gap between a feasible and an optimal schedule to handle the collision during the data transmission. In our proposed approach, first we find out two-hop neighbors of each node, then a particular slot is allotted to each node in order to prepare a feasible schedule using the RD-TDMA algorithm. Finally, the feasible schedule is fine tuned in a novel way to improve the efficiency in handling the collision by reducing the number of allotted slots. The proposed algorithm out performs the existing RD-TDMA algorithm in terms of number of slots required to handle the collision. The performance of the proposed protocol is carried out using Castalia simulator.**

***Index Terms*—Wireless Sensor Network, Media Access Control, TDMA, CSMA, feasible schedule, correlated contention**

I. Introduction

Wireless Sensor Networks (WSNs) consist of large number of independent sensor nodes which collects data from the surrounding environment, may process the data, and send the same to the base station(BS) using multi-hop communication. The BS ultimately sends the received data to the outside world.

WSNs are used in various fields starting from our day-to-day activities to critical real-time applications such as Home automation, Health Monitoring, Habitat Monitoring etc.

The source of power in each node of WSN is usually through battery, and it cannot be recharged from time to time as the WSN is mostly deployed in hostile environment. In WSNs, as the battery power is limited therefore one of the most important goal is to reduce the energy consumption which ultimately helps in prolonging the life time of the sensor networks.

In order to reduce the energy consumption, various sources of energy wastes in WSNs such as Idle Listening, Overhearing, Control Message Overhead, and Collision [1] [2] need to be handled in various ways.

Among the above mentioned sources of energy wastes, one of the important consideration in conserving energy for WSNs is reducing the collision during communication among the sensor nodes. Various MAC protocols [1], [2], [3], [4], [5], [6], [7] has been designed to reduce the collision during data transmission in WSNs.

Most of the existing slot scheduling algorithms for MAC protocols in WSN either prepare an optimal schedule to improve the bandwidth utilization of the channel or a feasible schedule in quick time to handle the collision during data transmission. In order to bridge the gap between the optimal and feasible schedule, in this paper, we propose a distributed slot scheduling algorithm which will prepare the schedule in quick time and also reduce the length of the schedule that minimizes the latency during data transmission. Initially, a feasible schedule is prepared using RD-TDMA [7] slot scheduling algorithm. Then, the number of allotted slots are reduced in a novel way to handle the collision and at the same time reduce the latency during data transmission.

The rest of the paper is organized as follows. Section II, briefly illustrates our proposed protocol. Section III, presents our experimental simulation results and its analysis. Section IV, finally concludes the paper.

II. Proposed Algorithm

Our proposed distributed slot scheduling algorithm prepares a TDMA schedule by going through various phases. In the first phase, two-hop neighbors of each node is calculated and the maximum of all the two-hop neighbors is found out. In the next phase, each node is allotted to a particular slot in such a way that collision can be handled during data communication by multiple nodes simultaneously. In the final phase, the number of allotted slots is reduced so that performance can be enhanced (i.e. reduction in latency during data transmission) at the same time collision can be handled during data communication.

A. Two-hop neighbor Discovery

[Step 1:] At the very beginning, each node broadcasts a one-hop neighbor-discovery(ND) message to its neighbors.
[Step 2:] The Nodes receiving this one-hop ND message generate a one-hop ND response message and send it back

978-1-5090-3315-7/16/$31.00 ©2016 IEEE

to the originator node which has generated the one-hop ND message.

[Step 3:] The originator node stores the node-id of all these nodes from which it has received the one-hop ND response message. These nodes forms the one-hop neighbors of the originator node.

[Step 4:] In due course of time, each node finds their one-hop neighbors. Then each node starts discovering their two-hop neighbors by broadcasting a two-hop ND message which contains it's own one-hop neighbors list.

[Step 5:] Nodes receiving the one-hop neighbor list populate it's two-hop neighbor list by adding the node id of the node from which it has received the two-hop ND message and the node ids present in the received one-hop neighbors list.

[Step 6:] After each node finds their two-hop neighbors, then they broadcast a max two-hop neighbors count message which contains the total number of two-hop neighbors present at their own ends.

[Step 7:] A node receives this message will check, whether the two-hop neighbors count at it's own end is less than the received one or not. If it is less then it updates it's max two-hop neighbors count and broadcasts a max two-hop neighbors count message in this network. Finally, the maximum two-hop neighbors count at all the nodes is found out which helps in preparing the number of slots to be present in a frame for the feasible schedule.

B. Allotment of slots

The number of slots in a frame is decided as per the maximum two-hop neighbors count. Each slot in a frame can be in one of the four states viz un-allotted, requested, granted, and allotted. The process for slot allotment proceeded as follows.

[Step 1:] Initially, each slot is assigned to a "un-allotted" state. Then, each node randomly selects a slot from the list of available slots and change the state of the chosen slot to be "requested". Finally, it broadcasts a slot allotment request message to their neighbors which contains the requested slot number to be allocated for it's own.

[Step 2:] Nodes receiving this slot allotment request message can either grant or reject the requested slot. The requested slot is granted if both of the following conditions are matched.

- The requested slot has not been granted by the receiving node to any other node.
- The requested slot has not been requested earlier by the receiving node itself.

[Step 3:] If any of the above condition is matched, then the receiving node updates the status of the slot to be "granted", and then broadcasts a slot grant message to it's neighbors.

[Step 4:] If a node that receives this slot grant message is the intended recipient, then it store the node id from which it has received the message.

[Step 5:] After a certain time period, the node checks whether it has received the slot grant message from all of it's one-hop neighbors or not. If it has received from all it's neighbors, then the requested slot is allocated to that node and the status of the slot is changed to "allotted". Finally, the node broadcasts a slot allotment success message to all of its neighbors. In case, the node has not received the slot grant message from all of it's one-hop neighbors, then it updates the status of the slot to "un-allotted" and broadcasts a slot allotment failure message to all of its neighbors, and then choose another slot randomly and continue from step-1 again.

[Step 6:] The nodes receiving the slot allotment success message updates the respective slot status to be "allotted". In case of failure, the nodes who have received the slot allotment failure message check if they have given grant to this failure slot earlier. If so, then these receiving nodes update that slot status at their own end to "un-allotted".

According to figure 1, the red color node randomly chooses the third slot and broadcasts a slot allotment request message to it's one hop neighbors. Then all it's one hop neighbors send back a slot grant message to it. After receiving the slot grant message for the requested slot from all of it's one hop neighbors, the red color node allocates the slot for him and broadcasts a slot allotment success message to it's neighbors. Finally, nodes receiving this message updates the status of that slot to be "allotted".

As per figure 2, the red color node requested the slot number three to be allocated for it and broadcasts a slot allotment request message to it's neighbors. Out of all the one hop neighbors, two of them have not sent back the slot grant message because the slot number three at these two nodes is already in the "allotted" state prior to the receive of slot allotment request message. After receiving the slot grant message, the red color node updates the status of the slot to be "un-allotted" as it did not receive the slot grant message from all of it's neighbors. Hence, it broadcasts a slot allotment failure message to it's neighbors. The nodes who have given grant after receiving the failure message updates the status of that slot to be "un-alloted".

C. Re-allotment of slots

In order to reduce the number of slots to be allotted, the number of allotted slots are reduced to half of the originally allotted slots. The process for slot re-allotment proceeded as follows.

[Step 1:] First the reduced number of slots is calculated (i.e. the half of the originally allotted slots). Then, each sensor node checks whether they can be reallotted to a new slot or not based on their originally allotted slot number and the calculated reduced number of slots. In case, the allotted slot number to the sensor node is greater than the calculated reduced number of slots, then the node allotted to the last slot is reallotted to the first slot, the last but one slot to the second slot and so on. After the convergence, all the nodes allotted to the last slot are reallotted to the first slot, the last but one slot to the second slot and so on.

[Step 2:] The nodes originally allotted to a slot are known as the owners of that slot. Due to the re-allotment, some nodes are reallotted to another slot and these reallotted nodes are known as the non-owner nodes of the newly allocated slot.

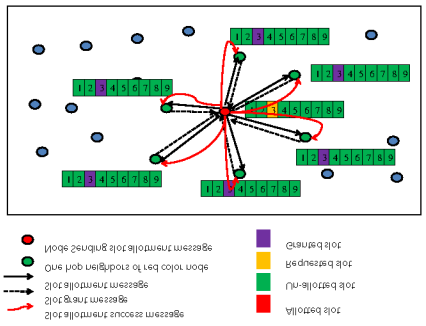

Fig. 1: Successful slot allotment

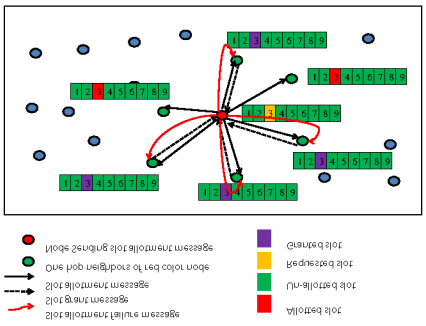

Fig. 2: Failure in slot allotment

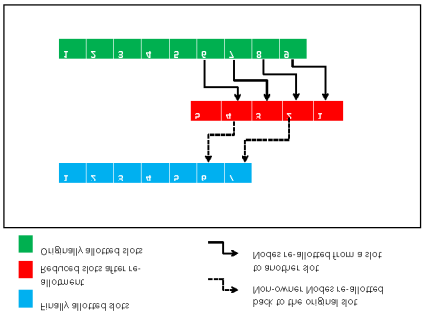

Fig. 3: Slot re-allotment

[Step 3:] After the re-allotment of slots, data transmission is carried-out to check which non-owner nodes are in collision during the same. In order to detect collision before data transmission each non-owner sensor node checks whether the medium is free or busy using CSMA. Once the node finds the medium is busy then collision will happen if the same node will transmit data along with the owner nodes.

[Step 4:] Hence, those non-owner nodes which are in collision are reallotted back to a new slot.

As per Figure 3, originally nine slots are allotted for the whole WSN to handle collision during data transmission. Then, the number of slots are reduced to five i.e. half of the originally allotted slots. During data transmission, out of the five allotted slots two of them i.e. slot number two and four are in collision. Finally, the reallotted nodes to slot number two and four are reverted back to two new slots i.e. six and seven. Hence, the final number of slots allotted to handle collision become seven.

The whole process has been carried out once at the beginning to prepare the schedule. This schedule will handle collision during simultaneous data transmission and the reduction in the length of the schedule (as compared to the feasible schedule) allows to transmit data with reduced latency.

III. SIMULATION EXPERIMENTS AND RESULTS

The performance of our proposed algorithm is carried out using Castalia simulator [8]. In this simulation, the nodes are deployed in uniform random manner. For this simulation, the number of nodes deployed are varied from 50 to 1000 with a fixed as well as varying density. Various sensor node parameters such as power level, energy consumption, transmission rate, etc. are taken into consideration as per the information given in cc2420 data sheet[9] and TelosB data sheet[10].

Figure 4 shows the average two-hop neighbors to the actual two-hop neighbors. Here, the average two-hop neighbors remain almost 50% of the actual two-hop neighbors. As the average two-hop neighbors are almost half of the actual two-hop neighbors, hence reduction of the originally allocated slots to it's half will improve the performance. Our proposed algorithm proves the same.

Figure 5 shows the number of slots in collision after re-allotment. Here, it shows that the number of slots in collision

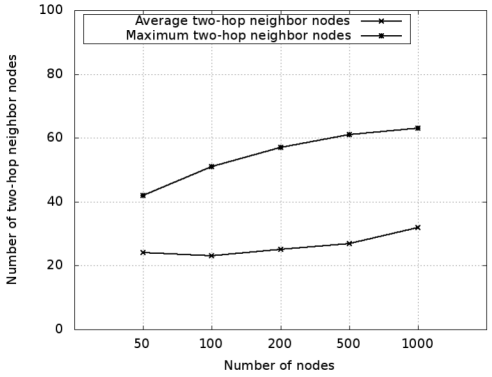

Fig. 4: With fixed node density, average two-hop neighbors compared to maximum two-hop neighbors

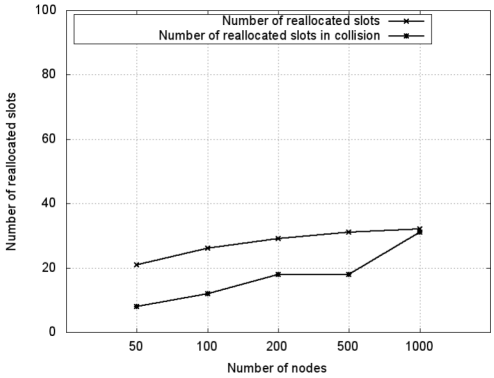

Fig. 5: With fixed node density, number of slots in collision after re-allotment as compared to the total number of slots after re-allotment

increases with the increase in number of sensor nodes. With increase in number of sensor nodes, the number of nodes allotted to a particular slot increases. This leads to the possibility of increase in collision due to the increase in number of sensor nodes.

Figure 6 shows a comparison of the number of allotted slots, re-allotted slots, and number of slots in collision after re-allotment. It shows that, the number of slots which are in collision almost remains the same with varying area. As

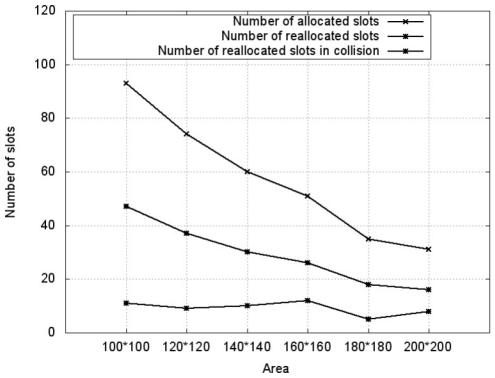

Fig. 6: With varying node density, comparison of the number of allotted slots, re-allotted slots, and slots in collision after re-allotment

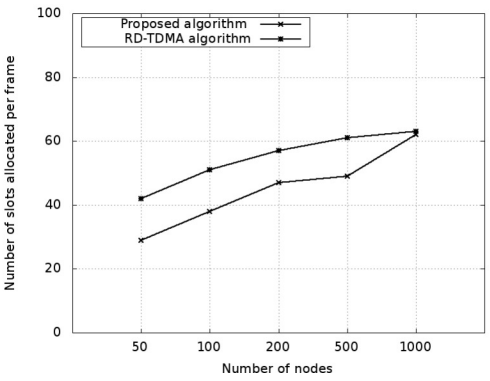

Fig. 8: With fixed node density, number of slots allocated in our proposed algorithm as compared to the RD-TDMA algorithm

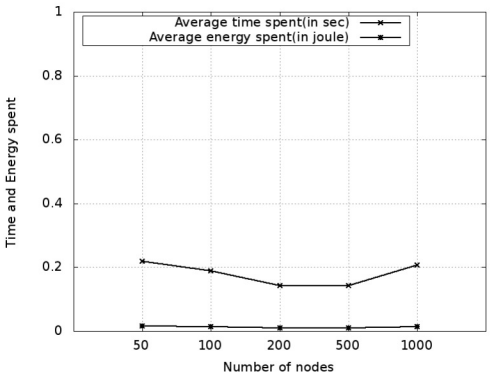

Fig. 7: With fixed node density, time and energy spent for re-allotment of slots

the number of nodes remains the same hence, the chance of collision also remains the same with varying number of nodes.

Figure 7 shows the time and energy spent for re-allotment of slots. This result shows that, the average time and energy spent almost remains the same as the number of node increases as our proposed algorithm is distributed in nature.

Figure 8 shows a comparison of the number of slots allotted in our proposed algorithm with the RD-TDMA to handle collision during data transmission. The results shows that, the number of slots allotted to handle the collision in our proposed algorithm is less as compared to the RD-TDMA algorithm i.e. our proposed algorithm performs better than RD-TDMA in terms of number of slots allotted to handle collision.

IV. Conclusion

In this paper, we proposed a distributed slot scheduling algorithm for hybrid MAC to handle collision during communication. In the first phase of the proposed algorithm, two-hop neighbors are calculated for every node and then maximum of all the two-hop neighbors is found out which will became the number of slots to be present in the feasible schedule. In the next phase, re-allotment of slots is done by reallocating the nodes from the last slot to the first slot, last but one slot to the second slots and so on. Finally, after re-allotment, the number of slots in collision is found out and accordingly the final schedule is prepared. This reduces the number of slots allocated for handling collision. The efficiency of the proposed algorithm has been evaluated with fixed as well as varying node density. Our proposed algorithm is compared with an existing distributed slot scheduling algorithm called RD-TDMA. This comparison shows that the proposed algorithm outperforms the RD-TDMA algorithm in terms of number of slots in the schedule to handle the collision.

References

[1] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the IEEE Infocom*, USC/Information Sciences Institute. New York, NY, USA: IEEE, June 2002, pp. 1567–1576.

[2] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 171–180.

[3] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-mac: a hybrid mac for wireless sensor networks." *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, 2008.

[4] I. Rhee, A. Warrier, M. Aia, J. Min, and M. Sichitiu, "Adaptive distributed randomized tdma scheduling for clustered wireless sensor networks," in *International Conference on Wireless Communications, Networking and Mobile Computing*, 2007, pp. 2688–2691.

[5] I. Slama, B. Jouaber, and D. Zeghlache, "Priority-based hybrid mac for energy efficiency in wireless sensor networks." *Wireless Sensor Network*, vol. 2, no. 10, pp. 755–767, 2010.

[6] S. Zhuo, Y.-Q. Song, Z. Wang, and Z. Wang, "Queue-mac: A queue-length aware hybrid csma/tdma mac protocol for providing dynamic adaptation to traffic and duty-cycle variation in wireless sensor networks," 2012.

[7] A. Bhatia and R. Hansdah, "Rd-tdma: A randomized distributed tdma scheduling for correlated contention in wsns," 2014.

[8] "Castalia a simulator for wireless sensor networks," *http://castalia.npc.nicta.com.au/pdfs/Castalia User Manual.pdf*.

[9] "Cc2420 data sheet," *http://www.stanford.edu/class/cs244e/papers/cc2420.pdf*.

[10] "Telosb data sheet," *http://www.xbow.com/Products/ Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf*.